**Homework Assignment 1**

Due date: September 10th, 11:55pm EST

**Problem 1**

A.)

a: Yes

b: Yes

c: No, because a Fruit may or may not be a Citrus

d: Yes

e: No, because a Citrus is not an Apple

f: Yes

B.) \* Assuming it is System.out.print, not System.out.println

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| X = 0.0  Y = 0.0 | X = -1.0  Y = 0.0 | X = 1.0  Y = 1.0 | X = 1.0  Y = -1.0 | X = 0.0  Y = 1.5 | X = 1.0  Y = 2.0 | X = 2.5  Y = 0.0 | X = -2.0  Y = 2.0 | X = 0.0  Y = 3.0 | X = -1.0  Y = -4.0 |

a: Fruit constructor calledCitrus constructor calledLemon constructor called

b: Fruit constructor calledApple constructor called

**Problem 2**

**Problem 3**

|  |
| --- |
| public static boolean removeDuplicates(ArrayList toChange)  {  boolean changed = false; // Used as a return value later on, initialized to false, and is changed to true if ArrayList is changed  // Iterates through the ArrayList, storing a word that will be used to go through the entire ArrayList again. This enables us to not care if the ArrayList is sorted or not, since it will go through the entire ArrayList to identify duplicates  for (int i = 0; i < toChange.size(); ++i)  {  String stored = toChange.get(i).toString();  for (int j = 0; j < toChange.size(); ++j)  {  // For each word that was stored, goes through the entire ArrayList to find duplicates  if (toChange.indexOf(stored) != toChange.lastIndexOf(stored))  {  toChange.remove(toChange.lastIndexOf(stored)); // Removes the duplicates  changed = true; // Changes the return value if it is changed  }  } // End for loop that searched the ArrayList for the stored word  } // End for loop that used all the stored words in the ArrayList  return changed;  } // End of removeDuplicates method |